얼굴인식을 통한 사람 얼굴 모자이크 처리 웹

팀명: 페이스 아웃

AI융합학부 20180000 길다영
20180000 송선영
20180000 최서윤

2021년 9월 7일

목 차

정의

- face recognition을 통해 지정된 사람을 제외한 다른 사람들의 얼굴을 모자이크 처리하는 시스템
- 다른 사용자들이 모자이크 처리 시스템을 이용할 수 있도록 웹 서비스 제작

목적

- 촬영된 영상에서 특정 인물 이외의 인물을 모자이크 가능
- 유튜버와 같은 크리에이터들이 많이 생겨나면서 (vlog와 같은 유튜브 영상에서) 미처 모자이크 처리되지 못한 사람들의 초상권 보호

"유튜브 영상에 내 얼굴이..." 보호 못 받는 SNS 초상권

학생 인기 직업 1위 열풍에
너도나도 '묻지마 생중계'
사생활 침해 등 주의했으면

알바생 10명 중 3명 "근무 중 원치 않게 사진·영상 찍혔다"
일부 가게, '노포토존' 선언
전문가 "개인 식별 유무에 따라 초상권 침해 여부 결정된다"

"늘어나는 개인 영상 촬영과 인증샷으로 초상권 침해가 꾸준히 증가하고 있다. 모자이크를 일일이 시도하기에는 많은 시간과 노력이 들고 미처 놓치는 부분이 발생하기 마련이다."

1
2
3

1. 수행 내용

Face Recognition System

- OpenCV을 활용해 얼굴을 트래킹할 것
- 얼굴이 특정 인물임을 인식할 것
- 특정 인물 외의 인물을 모자이크하는 기능을 추가할 것

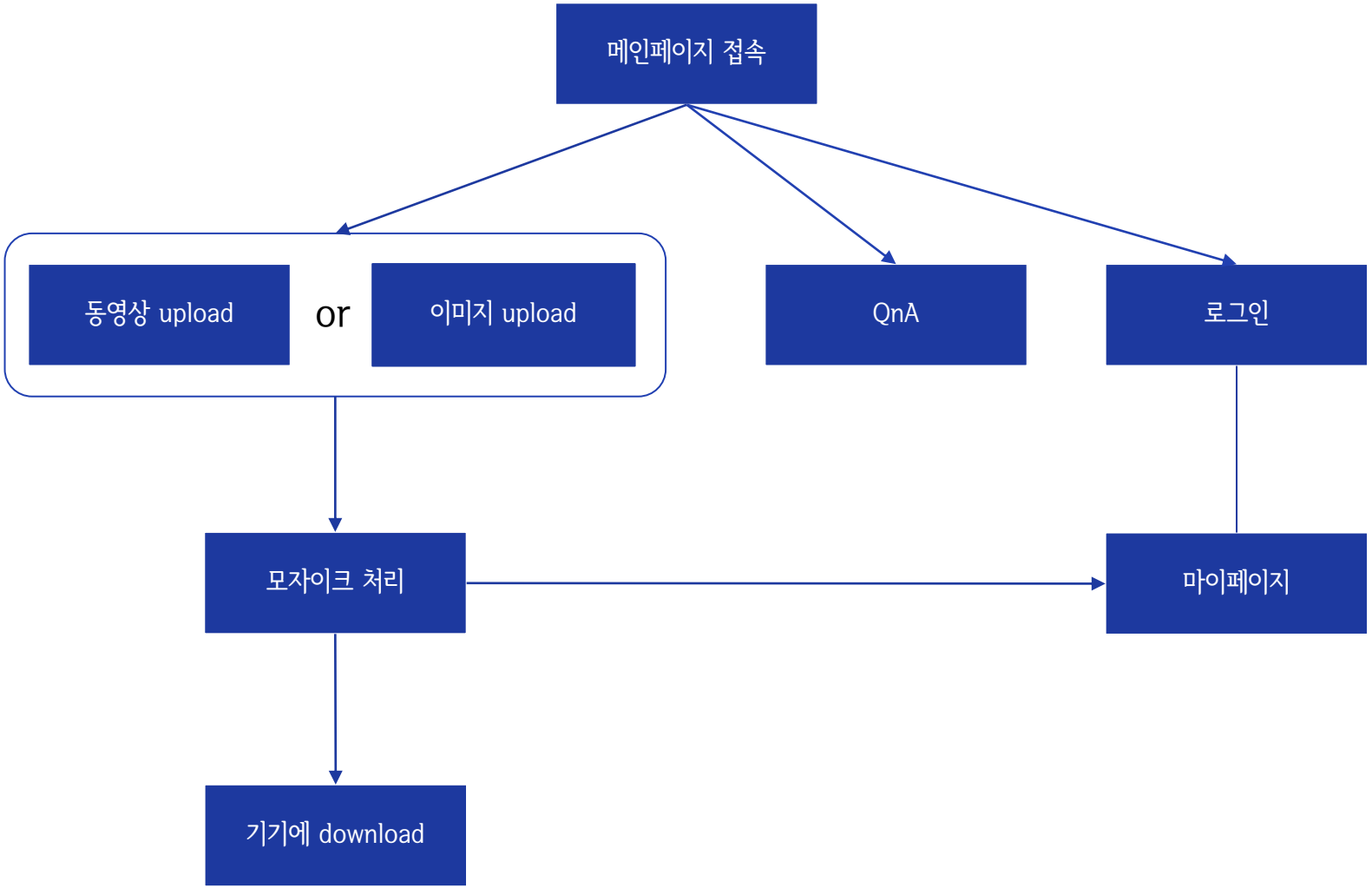
Web Service

- 동영상을 업로드 및 다운로드 받을 수 있는 기능을 추가할 것
- 얼굴인식 모델을 웹에 적용시킬 것
- 사용자의 계정으로 데이터를 보관하는 기능을 추가할 것
- 질의응답 게시판을 추가할 것

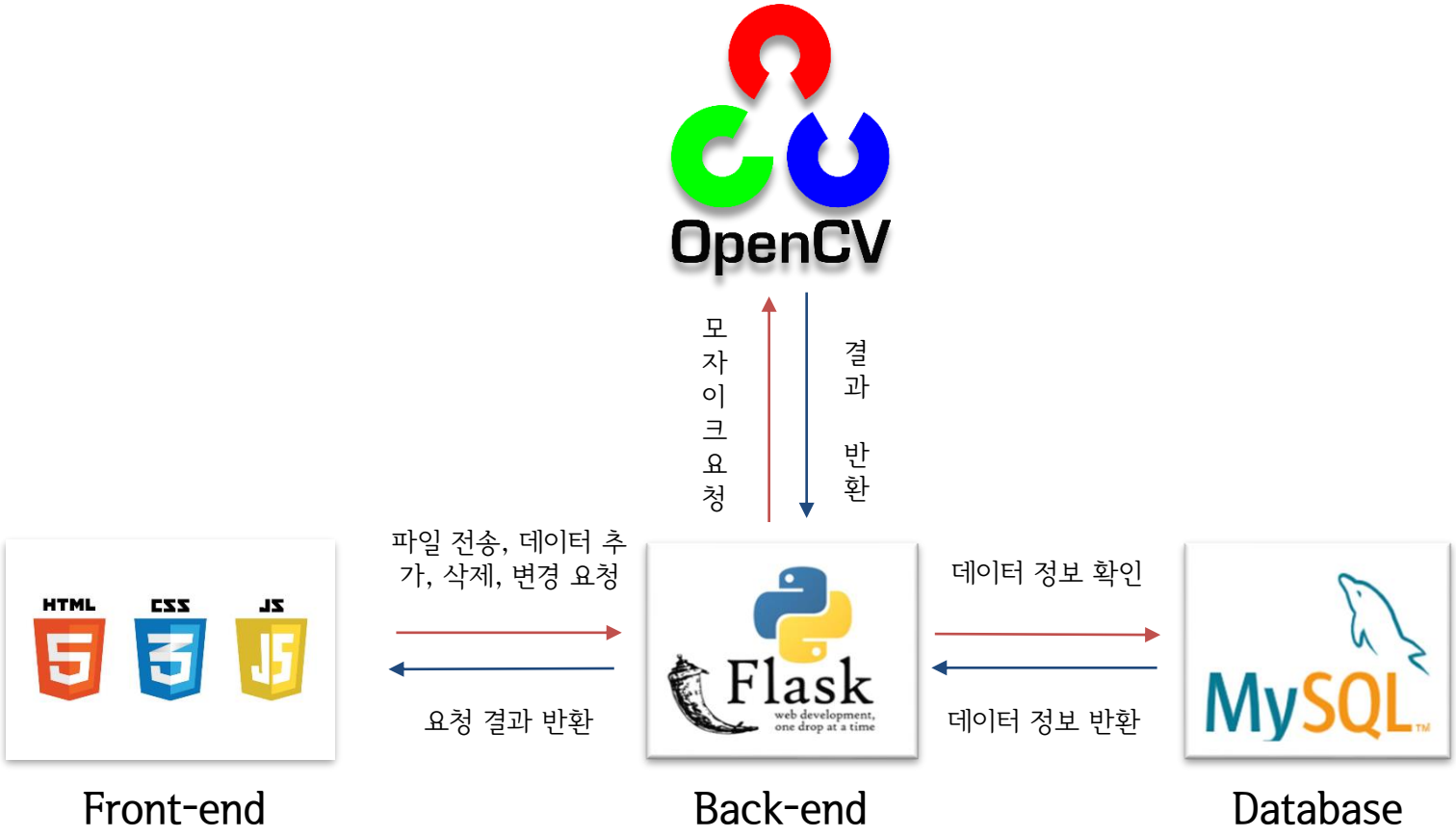
UI / UX

- 웹페이지 FrontEnd 부분을 단순하게 표현하여 가독성을 높일 것
- 회원과 비회원의 사용 기능을 다르게 할 것
- 웹사이트 사용 방법에 대한 설명을 적어 사용자들이 이용하기 쉽도록 할 것

2. 웹페이지 구현도


메인페이지 접속
동영상 upload
or
이미지 upload
QnA
로그인
모자이크 처리
마이페이지
기기에 download

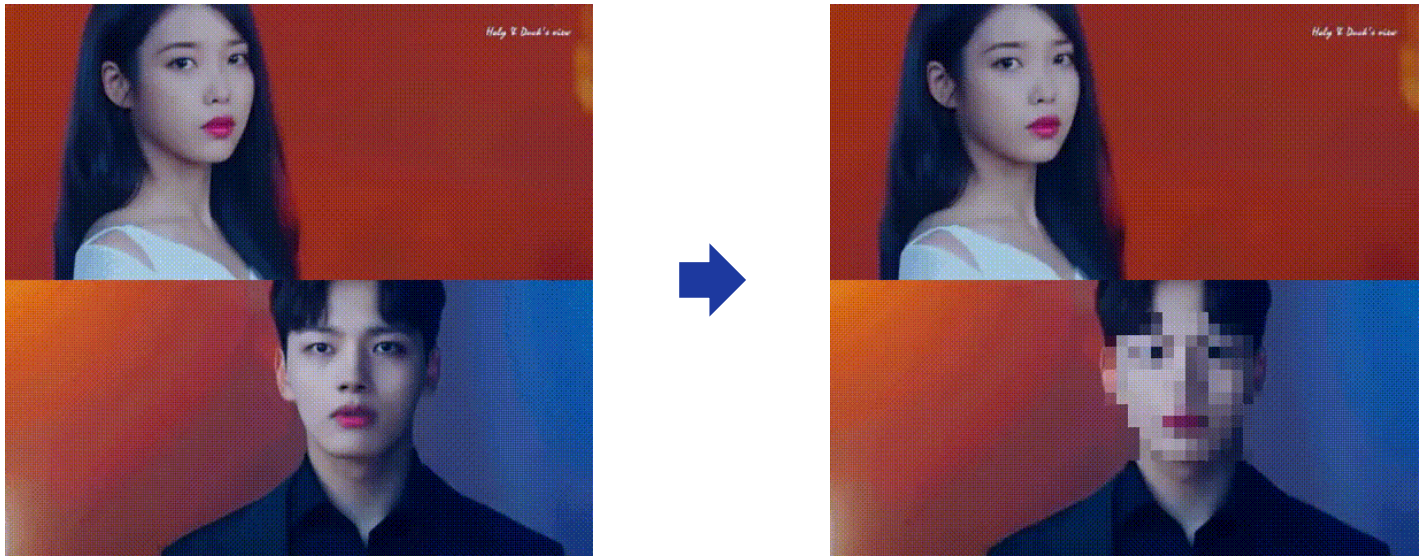
3. 시작품 구성도


OpenCV
모자이크 요청
결과 반환
파일 전송, 데이터 추가, 삭제, 변경 요청
요청 결과 반환
데이터 정보 확인
데이터 정보 반환
HTML
CSS
JS
Flask
web development, one drop at a time
MySQL
Front-end
Back-end
Database

4. 결과

<이미지 모자이크>

<동영상 모자이크>



4
5
6
7

1. Input image로 Face Recognition의 과정

Image 전처리 → Face detection → Face Comparison → 모자이크 처리

```
'''
STEP 1:
이미지를 꺼내서 RGB이미지로 변경하기
그리고 imshow로 출력해보기
'''

def input1(img_path):
    # 학습할 이미지를 RGB형태로 변경
    imgTrain = face_recognition.load_image_file(img_path)
    imgTrain = cv2.cvtColor(imgTrain, cv2.COLOR_BGR2RGB)
    return imgTrain


def input2(img_path):
    # 테스트할 이미지를 RGB형태로 변경
    imgTest = face_recognition.load_image_file(img_path)
    imgTest = cv2.cvtColor(imgTest, cv2.COLOR_BGR2RGB)
    return imgTest
```

학습할 이미지

모자이크할 이미지

‘학습할 이미지’와 ‘모자이크할 이미지’를 불러온다.
그리고 학습을 위해 파일을 RGB형식으로 바꿔준다.

4

5

6

7

1. Input image로 Face Recognition의 과정

Image 전처리 → Face detection → Face Comparison → 모자이크 처리

```
'''
STEP 2:
face detection = 얼굴 위치 확인
face encode = 그것을 숫자로 변경
'''

# 학습할 이미지
faceLoc = face_recognition.face_locations(imgTrain)[0]    # return (top, right, bottom, left), print로 확인 가능
encodeimg = face_recognition.face_encodings(imgTrain)[0]    # return 128 measurement
cv2.rectangle(imgTrain, (faceLoc[3], faceLoc[0]), (faceLoc[1], faceLoc[2]), (255, 0, 255), 2)
```

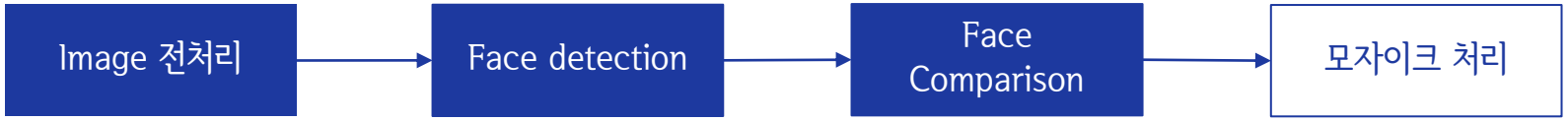
4

5

6

7

1. Input image로 Face Recognition의 과정




```
'''
STEP 3:
두 얼굴(의 인코딩 값)을 비교하고
두 얼굴(의 인코딩 값) 사이의 거리(얼마나 닮았나)를 비교

비교 모델: linear SVM
'''
results = []
faceDis = []
croppedFace = []
for encode_test in encodeTest:
    dic = face_recognition.face_distance([encodeIU], encode_test)
    if dic < 0.45:
        result = True
    else:
        result = False
    results.append(result)
    faceDis.append(dic)

faceLocTest = face_recognition.face_locations(imgTest)
for index, result in enumerate(results):
    face = faceLocTest[index]
    if result == False:  # 만약 result로 동인인물이 아니라고(False라고) 나오면
        imgTest = mosaic(imgTest, face)
return imgTest
```

‘학습할 이미지’의 인코딩 값과 ‘모자이크할 이미지’의 인코딩 값을 비교한다.
이를 distance비교라고 말한다. 얼굴이 서로 비슷할수록 작은 값이 나온다.

4
5
6
7

1. Input image로 Face Recognition의 과정

Image 전처리 → Face detection → Face Comparison → 모자이크 처리

```python
'''
STEP 4:
false인 부분 모자이크 처리
'''
def mosaic(src, face, ratio=0.1):
    (startX, startY) = face[3], face[0]    # left, top
    (endX, endY) = face[1], face[2]    # right, bottom

    face_img = src[startY:endY, startX:endX]    # [top:bottom, left:right]

    M = face_img.shape[0]
    N = face_img.shape[1]

    face_img = cv2.resize(face_img, None, fx=ratio, fy=ratio, interpolation=cv2.INTER_NEAREST)
    face_img = cv2.resize(face_img, (N, M), interpolation=cv2.INTER_NEAREST)

    src[startY:endY, startX:endX] = face_img

    return src
```

얼굴을 비교했을 때, distance가 0.45보다 크면
False로 지정하고 모자이크처리를 진행한다.

모자이크된 이미지

2. Input video를 Face Recognition 후, 모자이크 과정

이미지 및 동영상 전처리 → Face detection → 모자이크 처리

```
'''
STEP 1:
이미지를 꺼내서 RGB이미지로 변경하기
동영상을 꺼내기
그리고 imshow로 출력해보기
'''

def input1(video_path):
    # 학습할 이미지를 RGB형태로 변경
    imgTrain = face_recognition.load_image_file(video_path)
    imgTrain = cv2.cvtColor(imgTrain, cv2.COLOR_BGR2RGB)
    return imgTrain

def input2(video_path):
    # 테스트할 동영상을 불러옴
    test_video = video_path   # 테스트 비디오를 바꾸고싶다면 여기를 바꾸기
    cap=cv2.VideoCapture(test_video)
    return cap
```

학습할 이미지

모자이크할 동영상

‘학습할 이미지’와 ‘모자이크할 동영상’을 불러온다.
그리고 학습을 위해 파일을 RGB형식으로 바꿔준다.

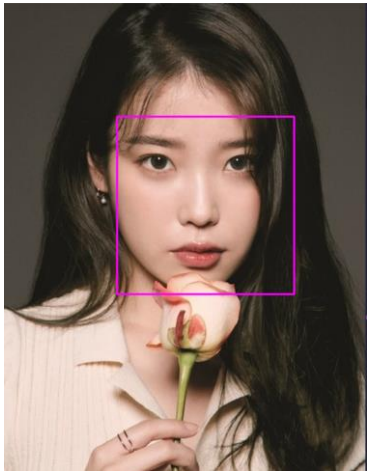
4
5
6
7

2. Input video를 Face Recognition 후, 모자이크 과정

이미지 및 동영상 전처리 → Face detection → 모자이크 처리

```
'''
STEP 2:
face detection = 얼굴 위치 확인
face encode = 그것을 숫자로 변경
'''

# 학습할 이미지
faceLoc = face_recognition.face_locations(imgTrain)[0]    # return (top, right, bottom, left), print로 확인 가능
encodeimg = face_recognition.face_encodings(imgTrain)[0]    # return 128 measurement
cv2.rectangle(imgTrain, (faceLoc[3], faceLoc[0]), (faceLoc[1], faceLoc[2]), (255, 0, 255), 2)
```



‘학습할 이미지’에서 얼굴을 인식한다.
인식한 얼굴을 encoding하여 숫자로 바꾼다.

2. Input video를 Face Recognition 후, 모자이크 과정

이미지 및 동영상 전처리 → Face detection → 모자이크 처리

```python
while (True):
    frame_num += 1
    print("현재 프레임:", frame_num)
    # frame 추출
    ret, frame = cap.read()  # ret: 잘 읽으면 True, 못읽으면 False # frame: 하나의 프레임
    if np.shape(frame) == ():  # 만약 더이상 frame이 읽히지 않으면 while문 빠져나오기
        break
    rgbframe = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    faces = face_recognition.face_encodings(rgbframe)  # frame에서 각 얼굴의 distance를 가져옴

    # 얼굴 거리비교
    results = []
    faceDis = []
    for face in faces:
        dic = face_recognition.face_distance([encodeimg], face)
        if dic < 0.4:
            results.append(True)
        else:
            results.append(False)
        faceDis.append(dic)

    # frame에서 얼굴이 True이면 네모치기, False이면 모자이크
    faceLocTest = face_recognition.face_locations(frame)
    for index, result in enumerate(results):
        index_face_location = faceLocTest[index]
        if result == False:
            mosaic(frame, index_face_location)

    # cv2.imshow("frame", frame)
    writer.write(frame)
    if frame_num == 1:
        cv2.imwrite(path2, frame)
    if cv2.waitKey(0) == 27:  # 만약 esc가 눌리면 while문을 벗어남
        break
```

모자이크된 동영상

동영상의 frame을 하나씩 보며, distance를 비교하여 0.4보다 크면 다른 얼굴로 인식한다.
그리고 이를 모자이크처리한다.

<기존 작품들>

이미지 모자이크 프로그램

1. 그림판 → 직접 손으로 모든 사람을 모자이크 처리해야 함

2. 포토스케이프 → 마우스로 지나간 곳만 모자이크 처리 됨
→ 어플을 따로 다운받아야 함

동영상 모자이크 프로그램

1. 유튜브 → 모자이크 처리할 사람을 일일이 지정해야 함

2. 곰믹스 → 특정 부분을 가리지 못하고
전체화면이 모자이크 처리 됨

<기존 작품들과의 차이점>

1.모자이크 처리 제외 사진 1장만 넣어도 자동으로 모자이크 처리가 된다.

2. 어플을 따로 다운받지 않아도 웹에서 서비스를 이용할 수 있다.

3. 홈페이지에 모자이크한 사진 또는 영상을 보관 가능하다.

4. 사람 얼굴이라는 특정부분만을 모자이크 처리할 수 있다.

5. 모자이크를 하고자 하는 사람이 다수일 경우, 한번의 클릭으로 빠르게 사람 얼굴을 모자이크할 수 있다.

<기대효과>

1. 초상권 보호

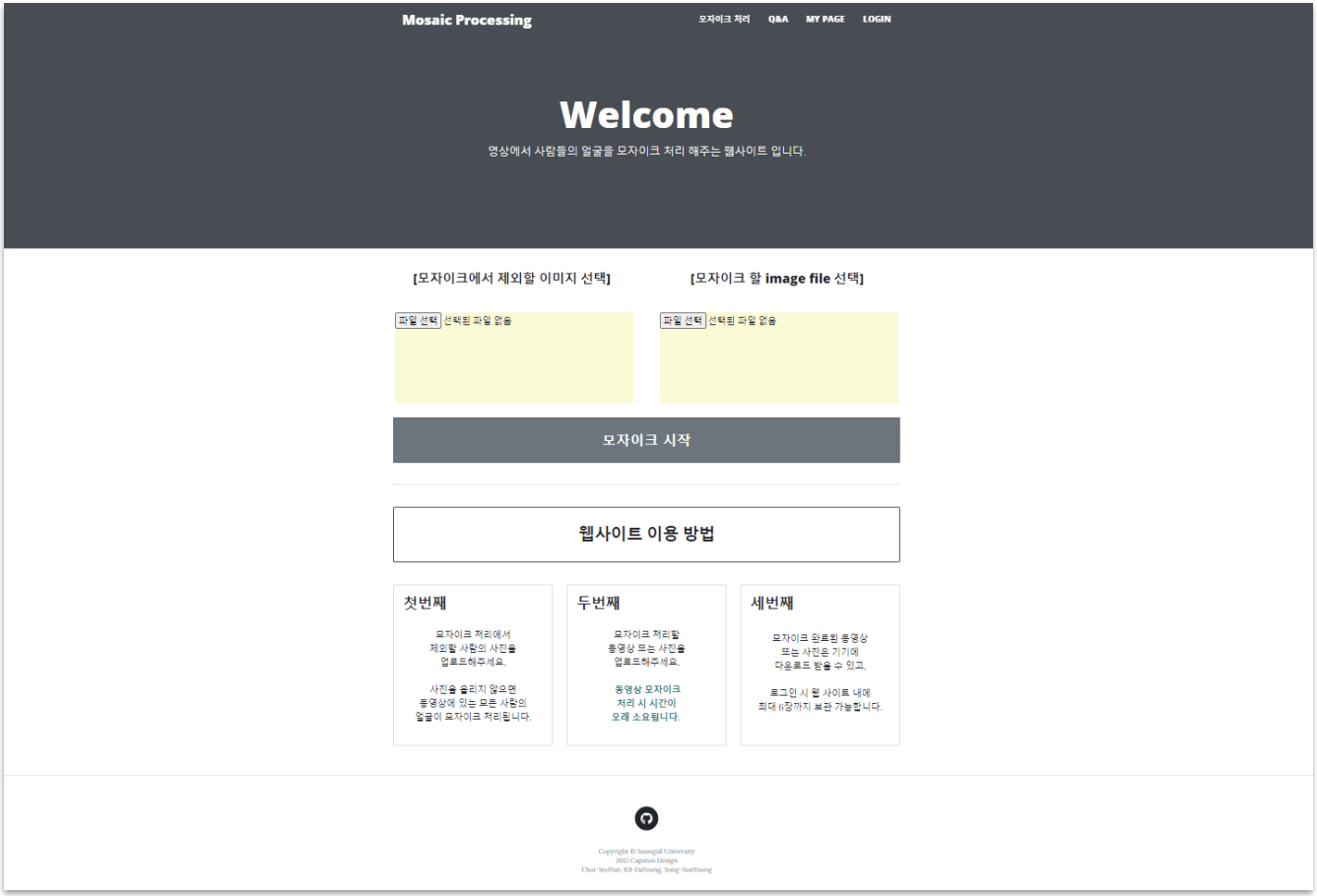
- 원치 않게 영상에 나온 사람들의 초상권을 지킬 수 있다.
- 사용자가 지정한 사람은 모자이크 처리에서 제외할 수 있다.

2. 기존보다 간단한 영상 속 모자이크 처리

- 영상 편집 전문가가 아니더라도, 간단하게 웹페이지에서 모자이크 처리를 할 수 있다.
- 로그인 시, 마이페이지에 영상을 보관할 수 있다.

<활용방안>

- SNS에 다른 사람의 얼굴이 나온 게시글을 올릴 때
- 개인 방송 또는 vlog와 같은 일반인이 나오는 영상을 올릴 때


Mosaic Processing
모자이크 처리
Q&A
MY PAGE
LOGIN
Welcome
영상에서 사람들의 얼굴을 모자이크 처리 해주는 웹사이트 입니다.
[모자이크에서 제외할 이미지 선택]
[모자이크 할 image file 선택]
파일 선택
선택된 파일 없음
파일 선택
선택된 파일 없음
모자이크 시작
웹사이트 이용 방법
첫번째
모자이크 처리에서
제외할 사람의 사진을
업로드해주세요.
사진을 올리지 않으면
동영상에 있는 모든 사람의
얼굴이 모자이크 처리됩니다.
두번째
모자이크 처리할
동영상 또는 사진을
업로드해주세요.
동영상 모자이크
처리 시 시간이
오래 소요됩니다.
세번째
모자이크 완료된 동영상
또는 사진은 기기에
다운로드 받을 수 있고,
로그인 시 웹 사이트 내에
최대 6장까지 보관 가능합니다.
Copyright © Soongsil University
2021 Capston Design
Choi-SeoYun, Kil-TaiYoung, Song-SunYoung

THANK YOU